Making Sense of Environmental Data in Real Time to Improve the Service Delivery in Assistive Technology

Pau Oliver i Farreny

Director: Joaquim Bellmunt
Image and Pervasive Access Lab
Supervisor: Dr. Javier Béjar
Universitat Politècnica de Catalunya

Bachelor's degree in Informatics Engineering

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya
October 2017

Abstract

The number of elderly people is rising globally. Many societies, including Western Europe, Singapore or Japan have ageing populations, facing predictions of inverted population pyramids in a few decades. Ambient Assisted Living (AAL) presents smart living environments aiming to provide a better assistance to dependent elderly people. It hopes to allow them to live independently at their houses, in a safe environment. UbiSmart is an AAL system that provides a web interface for caregivers and families to follow the real-time at-home behavior of the elderly. The aim of this thesis is to expand this system to add support for outdoor mobility. The work of this thesis spans the complete process of achieving a well-performing ($> 96\%$ accuracy) mobility classifier based on phone sensors. We developed an application to collect a labeled dataset; processed the collected data; trained multiple classifiers, tested and compared them; created a web microservice to deploy the chosen classifier; and integrated the service with the rest of the UbiSmart platform, adding support in the ontology for mobility detection. Finally, the resulting system has been deployed in IPAL's servers to be used in real environments in France and Singapore.

Resum

El nombre de població d'edat avançada està augmentant arreu del món. En moltes societats, com ara a l'Europa occidental, Singapur o el Japó, hi ha poblacions en envelliment, i s'enfronten a prediccions de piràmides invertides d'aquí a poques dècades. El concepte d'entorns de residència amb *assistència ambiental* (AAL en anglès) presenta entorns intel·ligents per a viure-hi, amb l'objectiu que persones grans puguin allargar la seva independència en un entorn segur. UbiSmart és un sistema AAL que posa una interfície web a disposició de cuidadors i familiars per a seguir, en temps real, el comportament a casa dels ancians al seu càrrec. L'ojectiu d'aquest projecte de fi de grau és ampliar aquest sistema per a afegir suport a mobilitat exterior. La tasca desenvolupada inclou el procés complet d'assolir un classificador amb bon rendiment ($> 96\%$ de precisió) basat en sensors de telèfons mòbils. S'ha desenvolupat una applicació per a col·lectar un dataset etiquetat, entrenat múltiples classificadors, els hem avaluat i comparat; hem creat un *microservice* per a publicar el model escollit; i s'ha integrat aquest amb la resta d'UbiSmart, afegint a l'ontologia el coneixement sobre el tipus de mobilitat detectat. Finalment, el sistema resultant ha estat desplegat als servidors web d'IPAL per a ser usat en entorns reals a França i Singapur.

Contents

Chapter 1

Introduction

1.1 Context

1.1.1 The problem

The ageing of the population is a phenomenon occurring worldwide. The World Population Ageing 2015 report [1] by the United Nations estimates that between 2015 and 2030 the number of people over 60 years of age will grow by 56%, going up to 1.4 billion people. This is not an isolated situation - virtually all countries are going to see an increase in number and proportion of old population. Furthermore, the same report projects that the number of people of very advanced ages (over 80) is going to be more than triplicated from 2015 to 2050.

The social impact of these demographic changes can become an extended problem if no effort is done to adapt to the needs of the situation. One of the problems that arises is the dependency of old people. After certain ages, people start needing external help to carry out ordinary tasks. As the age of populations increase, so does the amount of dependent people. When one reaches a level of dependency where an occasional caretaker is not enough, it is common to move to a nursing home. For people too dependent to cope with daily life activities alone (usually elders), nursing homes provide residential care, with skilled professionals available 24 hours a day.

However, nursing homes are not always the best answer, and there are a few reasons to delay the joining those care centers. First, they tend to be expensive, and they can become a big part or even more of the resident's income or pension. Second, the sentimental attachment to your own house –

not everyone is willing to move out of the place they spent most of their life unless it's strictly necessary, especially if there is family living with them. In addition, the path to dependence is often progressive. The level of assistance increases as one gets older, from needing some help getting out and walking the streets to cooking or getting dressed, before moving to a care center becomes the best option. Technology can help keeping track of the living conditions and try to predict risk factors. This age gap where the person is healthy enough to stay at home but needs certain control or care is the object of recent research and is the problem that this project tackles.

1.1.2 Ambient Assisted Living

An elder care survey to 127 people and their caregivers [2] shows that half of the surveyed people were worried about decreased autonomy, unsafe houses, cognitive problems, and the risk of accidents.

In this context of ageing population, ambient assistive is the concept of applying a context aware system to provide personalized and continuous care and assistance, dynamically adapted to their individual needs.

Active and Assisted Living Programme is a european funding activity that aims to create better life conditions for the elder and to strengthen the international industrial opportunities in the area of information and communication technology. As seen on its website [3], the specific aims of Ambient Assisted Living (AAL) are the following.

- extending the time people can live in their preferred environment by increasing their autonomy, self-confidence and mobility
- supporting the preservation of health and functional capabilities of the elderly, promoting a better and healthier lifestyle for individuals at risk
- enhancing security, preventing social isolation and supporting the preservation of the multifunctional network around the individual
- supporting carers, families and care organizations
- increasing the efficiency and productivity of used resources in the ageing societies.

1.1.3 IPAL

This project is going to be developed as part of the author's internship at the Image and Pervasive Access (IPAL). It is an international joint research laboratory established in Singapore since 1998 as an overseas laboratory of the French National Center for Scientific Research laboratory (CNRS) in order to enable collaborative research projects among CNRS, the National University of Singapore and the Institute for Infocomm Research of the Agency for Science, Technology and Research. Along the years, different institutions have joined the collaboration, latest renewed in 2011.

1.2 The solution

1.2.1 State of the Art

During the past few years, Ambient Assisted Living, AAL, has become a widespread topic, gaining more research involvement around it. Mokhtari et al. [4] presented in 2012 a review of the new trends to support people with dementia. According to the authors, the requirements of a system can be summarized in 4 requirements: remembering simple daily living tasks, maintaining their social links, feeling motivated to participate in everyday life and boosting their feeling of safety.

In 2004, Tapia et al. [5] did one of the first implementations, which they deployed in different residential environments with simple sensors looking for different activities. Depending on the activity, the accuracy rates were varied. In 2005, Helal et al. [6] presented an AAL prototype for houses, in which they divide the system in 6 layers: physical (hardware), sensor, service, knowledge, context-aware, and application. In 2011, Marinc et al. [7] presented an overview of the general requirements of an AAL platform to allow an efficient personalization. Summarizing them, the system needs to be non-invasive, invisible and stoppable. That same year, Morales et al. [8] presented the architecture and deployment of their AAL systems. The challenges they expose are personal tracking in order to localize a particular person with issues, human's willingness, skepticism and interaction. They also encourage to use ZigBee, a wireless communication with data flow control.

In 2014, Cubo et al [9] presented a complete AAl system hosted in the

cloud. They propose a platform to manage the integration and orchestration of heterogeneous devices, based on cloud computing technology. Their work on sensor integration inspired Bellmunt et al. [10] on their implementation of an AAL system for people with dementia. In 2015 they presented the challenges of a large scale deployment. They currently have systems running in multiple houses. The platform architecture of the system, UbiSmart, will be explained in Section 6.3.

1.2.2 Knowledge-Based Systems

A knowledge based system, or KBS, is a program that reasons and solves complex problems based on a knowledge base. KBS have an explicit representation of knowledge via ontologies, rules or similar tools, instead of implicitly via computer code.

In the context of AAL and sensor detection there are inaccuracies that should be taken into account. Inputs read from physical sensors, however, are not always fully reliable. Thus, a traditional rule-based system could fail to take into account inaccuracies, hardware failures or other problems. Aloulou et al. proposed in 2015 to take into account the confidence of measurements to handle uncertainty in semantic reasoning [11]. This is used in UbiSmart's model and will be considered in this project too. The inner workings of the KBS will be explained in further sections.

1.2.3 Fraility as an assistive metric

The existent pervasive health solutions computing medical values request a human intervention or focus on a specific medical issue. Bellmunt et al. [12] conceived a frailty model which uses sensor and environmental data to compute a multidimensional level of the frailty of an individual. Then, UbiFRAILTY was proposed, a frailty model embedded within an AAL UbiSmart framework. The framework adopted a minimal hardware invasion, relying on the developed software to infer the most accurate activity. The aim is to recognize aging activities of daily living through semantic and rule-based reasoning with a certain grade of uncertainty [11] through distant deployments. A final goal was to differentiate the quantity, the quality and the type of their daily activities.

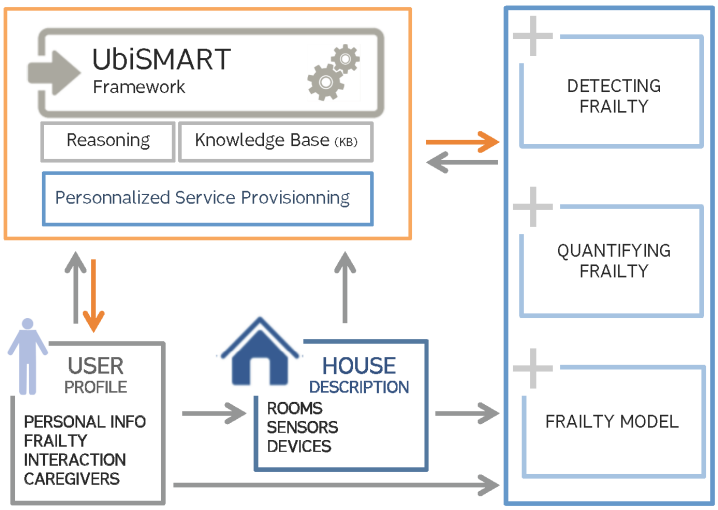

Figure 1.1: Frailty model integration through UBISmart [12].

1.2.4 Outdoors assisted living

The mentioned research was focused on tracking the activities of people inside their house or residence. This kind of assistance is getting more accurate and spread, but people also spend time outside, where no system of ambient sensors is available. However, their activities outside can help deliver a better assistance too, since they can be an indicator of the health conditions or even show possible incidents. We are particularly interested in the type of mobility the user engages in. We differentiate two types of personal mobility: Passive mobility: the user moves because he's using some means of transport - train, bus, taxi, etc. Active mobility: the user walks, runs or rides a bike - they are moving by their own effort. This would add one new element in the UBISMart environment currently shown in Figure 1.1: outdoors activity.

In order to detect these activities, we propose using a smartphone. Such devices are becoming more widespread in their use, expanding to older populations as well. A smartphone typically has accelerometer, gyroscope, magnetometer sensors and GPS.

1.2.5 Mobility Detection

The topic of activity classification based on on-body sensors has been previously explored with very positive results. In [13], Mannini *et al.* present a

dataset obtained from a network of five on-body accelerometers, used with different classifiers. A 98.5% accuracy is achieved with a geometric approach, Nearest Mean, while a probabilistic approach, Naive Bayes, yields a 97.4% accuracy. They demonstrate that a Markov modelling can be used to further increase the accuracy, to 99.1% using a chained hidden Markov model and after rejection of spurious data. However, those studies were performed in controlled scenarios allowing the researchers to define the position of the device and duration of the records. More recently, Tedesco *et al.*[14] presented a complete study on a mobility classifier. In this case the authors restricted the position of the phone at the moment of recording the activity. In our case, we are interested in a very light real deployment model, which is able to identify the type of mobility from smartphone sensors without too many further constraints or laboratory conditions.

1.3 Stakeholders

- Elderly people with mild dementia or assistance needs. They will be the recipients of the app, and the project is focused on improving the quality of their lives. However, the use that they will give to the system should be passive, with no need for a regular interaction with their phone or sensors used.
- Assistance professionals or caretakers. The benefits of tracking the activities of the patients will help professionals identify their needs and some changes in their health.
- Patient's family. Not only the professionals but also the patient's family will be able to make use of the information available and the analysis performed by the system in order to help the patient improve their daily life.

1.4 Scope

The aim of the project is to develop a way to expand the AAL outside the house.

We will identify what kind of information can be derived from a phone's sensor readings. By creating an Android app we will be able to collect sensor

information, and then create a dataset to work on. This will let us study the different patterns when users are outside. Machine learning models will be trained on the data and evaluated. We will deploy the best model to a web accessible server so that it can be used as an API. Expanding the existing ontology and knowledge based system, we will be able to expand the current knowledge of the system and hopefully improve the quality of the assistance. The system will be integrated with the UBISmart platform, so integration with the cloud platform will be required.

In section 2.4 we will explore the possible obstacles that we may face and the actions to avoid or fix them.

1.5 Methodology

1.5.1 Waterfall model

The project will be developed in a relatively limited time. This time might be too short to follow a pure agile methodology with many iterations. Furthermore, agile is optimized for teams, while this project is basically going to be developed by one person. Thus the main methodology will follow the waterfall model, where progress flows steadily through the different phases. For some of the subtasks though, we will incorporate elements of scrum. This project is going to be part of an existing project and I will work closely with their team. Some tasks won't have specified requirements until later on, and these may change during the development of the project, especially the integration with the server and cloud platform. Elements like daily scrums (stand-ups) will most likely take place in those cases too.

1.5.2 GitHub

The team at IPAL uses git as a version control software, with GitHub acting as the remote repository and code hosting service. This project will use git and GitHub too, following the expected programming workflows: opening and solving issues, with different branches per functionality, creating and reviewing pull requests, etc.

1.5.3 TDD

Test driven development is a development process where requirements are turned into specific test cases. In test driven development, you first write tests for the feature before starting programming it. The tests need to be exhaustive so that passing the test means meeting the requirements for the feature. We will use test driven development as much as possible. In some cases the testing phase won't be automated (it might need a user to walk with their phone and see if the system recognizes the activity, for example). However a goal is to be able to test all the code written. This matches the current workflow of the team, where through code reviews and tests a teammate validates that a feature matches the expected behavior.

1.5.4 Changes

The TDD method, even if useful, didn't take a big role. For data analysis, a good way of visualizing the steps in the experiments is with Jupyter Notebooks, that allowed to create *literate* python code commented with Markdown text in order to make it easily reproducible and modifiable. For the app development, the relatively small size of the apps made it possible to test them from the user perspective without too much trouble. The features of these apps was rather limited and simple so the benefit from automated tests would not have been too big. Other than this, no changes were done in the methodology.

Chapter 2

Project planning

2.1 Schedule

2.1.1 Estimated project duration

This section will cover the time planning of the project, describing the tasks to be executed during the project and timeline of the dependencies between tasks.

The estimated project duration is approximately 6 months. The internship, and thus the main project, starts on April 1st, 2017 and will be finished on September 30th, 2017. The Project Management course (GEP) happens throughout March, and the final presentation will take place in October.

2.1.2 Considerations

The first version of this plan was open to modifications. When presented for the GEP course, we noted that it might be changed during the project to account for any changes or unexpected circumstances of the project. Indeed, this plan has been modified since it was first presented. Later on in this section we explain the changes and their consequences.

2.2 Project planning

2.2.1 Project Management course

Before anything else, a course on Project Management (GEP) will be taken. The course has two objectives. First, to introduce the methods and tools

that will enable students to prepare their theses successfully. Second, to provide students with a solid foundation in the literature on the subject, in order to develop the skills and abilities that will be used in their professional activity.

During the course, the following aspects of the project will be defined:

- Project context and scope
- State of the art
- Project planning
- Project budget
- Sustainability and social commitment

2.2.2 Problem analysis and solution design

The goal of this phase is to analyze the project formulation and identify the requirements, needs, and features that our solution has to address, and then design a solution.

To achieve the goal, we will carry out an analysis of the problem, a study of the platform where it needs to be integrated, we will consider the state of the art and then design a solution that solves the problem while taking into account all these aspects.

2.2.3 Project development

This phase is the execution of the main task: the development of the project itself. It covers all the tasks related to the development and testing of the algorithms and app.

Analysis of existent data

What does the UBISmart platform tell us? An analysis of the kind of data available to see what information we can extract, if any, that tells us about the outdoor activities.

First expansion of the existent ontology

With the insight gained after the first analysis, ways of expanding the ontology will be considered. Can we infer anything about the outdoor activities of the user by exploiting the data collected by the current sensors? Then the ontology behind the system will be expanded to reflect that knowledge.

Dataset definition

What information can we extract from an Android app that would be useful to understand the user's mobility patterns? The needed sensor data and the best way and conditions of recollection will be defined.

App creation, data recollection

Development of a smartphone app to collect the data required as an initial dataset. The app will be used to generate data to be used as a ground truth and training data.

Data analysis and prediction

Different algorithms and machine learning techniques will be considered in order to create a way of classifying the mobility in active or passive and, if possible, which specific type.

Extension of the ontology and creation of new rules

In order to integrate with the knowledge-based system used at IPAL, we will design new parts to add to the ontology and create new rules that incorporate the knowledge related to outdoor activities collected by the developed app.

Integration with the UBISmart API

The proposed algorithm(s) and prediction will be integrated to the existing HTTP API in order to make it part of the ecosystem developed at IPAL which other systems can use through an API.

End-user app proof of concept

A simple smartphone app will be developed to show the operation of the system. The app will collect sensor data periodically and sent to the web server mentioned in the previous point, so that the classifier can be tested with real time data.

Testing

Validating the solution and its implementation. This will be an ongoing process during the development of the project, starting as early as possible and using test-driven development when the context allows it. The goal of starting early is that any defects in the requirements or design phase are captured in early stages.

Project memory

A document explaining all the project development process will be written.

Oral presentation

Task	Expected duration (h)
Project Management Course	80
Analysis and solution design	90
Expansion of ontology	10
Initial data analysis	20
Dataset definition	15
App creation	80
Data analysis and prediction	200
Ontology and rules design	50
Integration with the API	90
User app development	40
Testing and polishing	60
Project memory	50
Final presentation	15
Total	**800**

Table 2.1: Estimated time involvement

A final presentation will be prepared to orally explain the project to an evaluation jury.

2.3 Estimated time involvement

Table 2.1 shows a timetable of the estimated time to be used for each task. As seen there, the total expected time dedication is of around 800 hours. In order to finish it in 23 weeks, the student will need to work $\frac{800h}{23weeks} \approx 35h/week$ during the project.

2.3.1 Gannt diagram

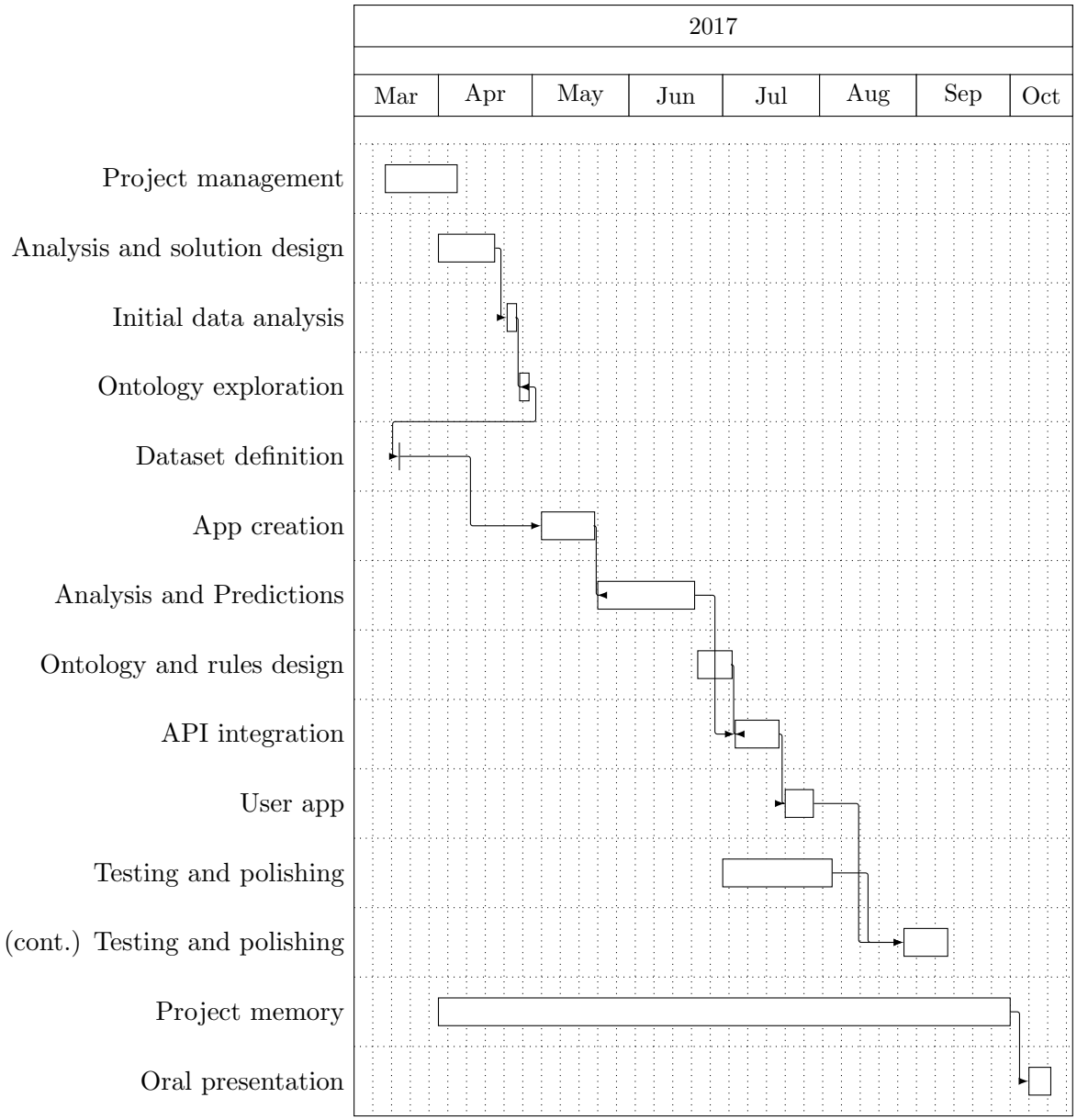

Figure 2.1: Planning timeline

2.4 Possible obstacles and solutions

As we mentioned, even though the plan presented here has been detailed carefully and as detailed as possible at the moment of the writing, it is open

to possible changes. The frequent meetings with the project directors and the agile approach to the development of the programming subtasks will allow to detect any incoherence and propose a plan change in time.

2.4.1 Dataset design

A bad dataset design could lead us to a frustrating data analysis. The data collected through the designed Android app will be used for the analysis of how to classify different types of mobility. If the data collected is too independent from the type of mobility, it will be difficult to use for our goals. The main strategy to overcome this danger is a close supervision by the project directors, with experience in the field, and a careful selection of what data to collect, probably choosing a broader selection than strictly needed in order to later discard the noise instead of facing the described problem.

2.4.2 Timeline

This project starts with a delay of the begin date. Any further delays, for example on the visa application procedure, can become a serious problem.

This is a completely external factor which all the involved parts tried to avoid, but a delay happened nonetheless and the internship started no sooner than April. As we planned in the beginning, we solve the issue by revising the time plan and moving the presentation date to October.

2.4.3 Insufficient data

As we explained, we will collect a dataset ourselves. This will limit the amount and variety of the data available and will be a restriction to keep in mind when doing the analysis. For example, deep learning usually requires large or very large datasets, and hence might not be a feasible technique to use in this project. The way to overcome this is to choose carefully the algorithms used, ask the expert directors for a revision of the ones we choose, and apply proper generalization testing – that is, making sure that the models developed are able to generalize and are not overfitting to the data. Common ways to achieve that are splitting the dataset in test and training sets or using cross-validation.

2.4.4 Analysis interpretation

It is often tricky to understand the output of statistical tests and machine learning models. There are, very often, biases to consider, and interpreting the meaning of the trained model parameters is not an easy task. To avoid any errors in the analysis, the methods used will follow the expected rigor as studied in the university courses of Machine Learning and Applied Statistics by the author.

2.5 Revised timeline

The first project plan, written during the GEP course, assumed a start date of beginning-mid March. For unavoidable reasons, the start of the project was delayed until April, so we acted like described in 2.4.2 – redoing the time plan considering an April - September project.

Considering the length of the project became longer, we allotted some more time in the biggest tasks. This will allow to achieve a better level of completion of each task, with a lower risk and hopefully a higher quality making the results deployable to the real system. After a research of the state of the art, we increased considerably the time for developing the model. It's a rather big task which includes the experiment design, some literature reading, many tests and tries and understanding different algorithms.

In addition, we decided to create a small smartphone app that demonstrates the use of the mobility classifier. This app development time is not expected to be high since we will have the experience and the code of the previously developed app to collect a labeled dataset, so some challenges with the reading of sensor values will have already been solved.

Chapter 3

Economic plan

In an engineering project, cost management is necessary to ensure that the project is undertaken within the limits of the budget approved by the client, as studied in the project management course (GEP).

Even if this project is developed as a degree thesis, all the resources and hours spent have a price, and the accumulation of all those represents the project cost. The goal of this chapter is to present an estimation of the cost of this project.

3.1 Hardware resources

This project will be developed on a desktop computer, with its corresponding monitor, mouse and keyboard. An Android device will be needed to build the data collection app and use gather the dataset.

Table 3.1 shows the hardward budget. The price and amortized cost are shown in euros (€) and the useful life is measured in years. The amortized cost is computed as $\frac{P * U * H}{L}$, where H is the number of hours of the project (i.e., 800) and L is the total lifespan in hours, considering 8h/day workdays and that there are approximately 1005 working days in the next 4 years and 752 working days in the next 3 years.

Hardware	Price (P)	Units (U)	Lifespan (Y)	Amortized Cost
Desktop computer	900	1	4	89,55
Mouse and Keyboard	70	1	4	6,97
OnePlus 3T	440	1	3	58,51
Total	1410			155,03

Table 3.1: Estimated hardware cost

3.2 Software resources

The software used for this project will be: Atom, Vim, Python, Anaconda, Android Studio, LaTeX, nodejs, npm and git. The licenses of all these packages allow us to use them for free, so the estimated price of the software required is 0 €.

3.3 Human resources

Since this project will be developed by only one person, the author will need to take different roles at each different situation. The different roles and expected payment for each of them are shown in table 3.2. According to that, table 3.3 shows the time spent by each role on each of the tasks, the estimated cost of the task computed and the total human resources cost. The cost of each task is computed as

$$\sum_{i \in R} C_i * H_i$$

where R is the set of the three different roles, C_i is the hourly payment for the role i and H_i is the number of hours of role i spent on that task.

Role	Hourly rate (€/h)
Software Engineer	35.00
Project Manager	50.00
Software Developer in Test	30.00

Table 3.2: Hourly payment per role

Task	Project Manager	Software Engineer	Software Test Developer	Cost
Project Management Course	80	0	0	4000
Analysis and solution design	30	60	0	3600
Expansion of ontology	0	10	0	350
Initial data analysis	0	20	0	700
Dataset definition	0	10	0	300
App creation	0	65	15	2725
Data analysis and prediction	0	180	20	6900
Ontology and rules design	0	45	5	1725
Integration with the API	0	60	20	2700
User App	0	20	10	1000
Testing and polishing	0	0	60	1800
Project memory	50	0	0	2500
Final presentation	15	0	0	750
Total	175	470	130	**29100**

Table 3.3: Distribution of hours per role and task; cost of each task and total cost of human resources

3.4 Other resources

3.4.1 Electricity

Electric power will be needed to use the desktop computer and to charge the mobile phone. Given a combined consumption of 175 W and a price of 0.13 €per kWh, the estimated total cost of the required electricity for the 800h is 18.20 €.

3.4.2 Internet connection

An internet connection will be needed throughout the project. At an approximate cost of 50€/month for a fast internet connection, the approximate total cost is 300 €.

3.5 Total

Table 3.4 shows the total economic cost of the project. A 10% of the total cost is added to account for any possible contingencies during the project.

Resource	**Cost (€)**
Hardware	155,03
Software	0
Human	29100
Electricity	18.20
Internet	300.00
Subtotal	29573.23
Contingency 10%	2957.32
Total	**32530.55**

Table 3.4: Total cost of the project

3.6 Revised budget

The budget presented here differs from the first version written before the start of the project. As explained in the previous chapter, an important change of timeline took place, and presenting in October instead of June meant an time extension of 30%. The total project cost of that first plan

was of 26242.79€. The increase of the budget is justified by the increase of time involvement.

Chapter 4

Sustainability

In the book *Change Everything: Creating an Economy for the Common Good* [15], Christian Felber presents a type of economy where businesses follow an approach to integrate with the world they live in, taking into account the impact that they have. This section presents a basic sustainability report based on the method suggested by Felber.

4.1 Social impact analysis

This project is focused on dependent people: ageing population or people with dementia. The sensors in their house lets the caretakers and family to follow an accurate status evolution of the user. The ontology and classification behind the system allows to describe the situation of the person. By expanding the tracking to the outside activities, an even more complete report can be provided.

This has a positive impact on the user's life, easing the health check process by automating parts of it, giving the ability to caretakers to provide a better targeted attention, in the right timing and ammount, and with the potential of improving the user's health.

The direct impact is not going to be multitudinary right after the completion of this project, since that depends on the deployment of the system to real users, families or nursing houses. We expect that it will first be implemented for the pilot users currently using the existing UBISmart system.

4.2 Economic analysis

In the previous section we showed an economic analysis of the project, showing its cost structure and use of resources. These resources will be used with care.

The project has currently a goal more focused on research than on immediate returns, so at this stage there is no plan for a direct economic benefit from it. However, we predict that this project would add interesting features to a commercial version of the UBISmart platform, making it more desirable to the potential users.

4.3 Environmental impact analysis

In this section we look at the environmental impact of two aspects of the project: its development, and the deployment of the resulting app and system.

The environmental impact of the deployment of the results is rather low. If we assume that the average person carries a smartphone with them most of the time, having one more app installed wouldn't add a significant energy consumption. As for the in-home sensors and the servers, those are installed already for the current solution and our contributions will be in the form of software added to them.

The development of the project will suppose around 107 kWh of electricity, considering the power consumption seen in section 3.4.1. This can be translated to approximately 53.44 kg of CO2 according to [16]. The hardware used will have to be removed at some point in the future. When that happens, the devices will have to be broken apart, hopefully recovering some recyclable or reusable components, but most of them will become different contaminants and have a negative impact on the environment.

Chapter 5

Laws and regulations

We have to make sure that this project complies with the laws and regulations that affect it. This project is carried out in Singapore, and the regulations that affect us are the Data Protection Act and an ethical approval needed at IPAL.

5.1 Personal Data Protection Act

The Personal Data Protection Act, which came into full effect on 2nd July 2014, wants to address concerns from individuals about how their personal data is being used, while aiming to "strengthen and entrench Singapore's competitiveness and position as a trusted, world-class hub for businesses" [17]. The overview of the act summarizes it in the following way.

> The PDPA establishes a data protection law that comprises various rules governing the collection, use, disclosure and care of personal data. It recognises both the rights of individuals to protect their personal data, including rights of access and correction, and the needs of organisations to collect, use or disclose personal data for legitimate and reasonable purposes. ([17])

Next we show a guideline to comply with the different points of the act. The data that will be collected throughout this project is a labeled dataset of phone sensor readings and the activity the person was doing. The users involved are all members of our team or people in direct contact with them.

5.1.1 Consent obligation

- **DO** Seek consent before collecting, using or disclosing personal data, and allow individuals to withdraw consent.
- **DO** Notify individuals of the purpose of collection, use and disclosure of their personal data so that they can give their consent.
- **DON'T** Mislead, deceive individuals into providing their personal data or force/coerce them into providing consent.
- **DON'T** Use existing personal data for new purposes without collecting fresh consent.

All our users are and will be informed of the purposes of the data, these points will be all followed.

5.1.2 Purpose limitation obligation

- **DO** Restrict use of an individual's personal data to the purposes for which it was collected, used or disclosed.

The activity data that we collect has the only use of analysis and training activity classifiers.

5.1.3 Notification obligation

- **DO** Notify individuals of the purposes before collecting, using or disclosing their personal data.
- **DON'T** "Use first" and "notify later".

The app through which we will collect the data won't be published to an app store. We will have to install it manually to anyone who can help us collect sensor data, and they will be notified first.

5.1.4 Access and correction obligation

- **DO** Facilitate individual's request to access or correct their personal data within 30 days.

- **DON'T** Obstruct individual's access or correcction of their personal data.

We won't probably have any requests to modify data, since the "personal" information comes from sensors (in contrast to being, for example, email addresses, phone numbers or friend lists). But if any of the participants requests a change (to, for example, the location visited at a certain date), we will facilitate it – and analyze if the change affects the data analysis.

5.1.5 Accuracy obligation

- **DO** Make effort to ensure personal data collected for decision making is accurate and complete.
- **DON'T** Collect or retain personal data that you don't need.

The data collected from smartphones is transmitted to a server and stored as-is, with no alterations. Any personal data is as accurate as the sensors can provide.

5.1.6 Protection obligation

- **DO** Do encrypt personal data carried on mobile storage devices.
- **DON'T** Don't disclose personal data to anyone without verifying the individuals identity.

The personal hard drive of the author's laptop is encrypted. No personal data is carried on other mobile storage devices, and no data will be disclosed without warning.

5.1.7 Retention limitation obligation

- **DO** Destroy or delete personal data that is no longer required.
- **DO** Have a protocol to check the relevance of personal data periodically.
- **DON'T** Retain personal data if it no longer serves any legal or business purpose.

The dataset collected might be useful for some time after the completion of this project, to improve or retrain the models. Every time such an improvement is done, the relevance of the data of the current users can be checked. The non-relevant data can then be be deleted if there is no use for it.

5.1.8 Transfer limitation obligation

This section is about transfers of data to offshore companies. No such transfers will be carried out in our case.

5.1.9 Do Not Call Registry

The Personal Data Protection Act establishes the obligation of keeping a Do Not Call Registry, to contact individuals only if their consent is clear. Our personal data does not include contact information and we are not going to use it to contact any individuals.

5.2 Institutional Review Board

IPAL, as part of the CNRS (Centre national de la recherche scientifique), follows the French and Singaporean legislations in all applicable regards. There are test deployments in place right now in both countries, with the sensors placed the apartments of the users providing real data.

In France, there is the role of "Data Protection Correspondent" – a person responsible for ensuring compliance with the law withing the organization, which simplifies the treatment of the personal data. A pilot site in Montpellier is following the procedure established by the "Persons Protection Committee" (CPP), which looks after the protection of the participants to medical research and the compliance with the legislation in the context of medical research.

For the pilot site in Singapore, CNRS is following the procedure established by the National University of Singapore (NUS) Institutional Review Board (IRB). The IRB reviews all research with human participants before the research starts. It monitors it via feedback received and regular reports. The NUS-IRB follows the Singapore Good Clinical Practice guidelines among others. The pilot site by IPAL taking place in Singapore has

the ethical approval of NUS-IRB.

If we treat any data belonging to the French or Singaporean research tests, we will do so taking into account these guidelines and legislations.

Chapter 6

Ambient Assisted Living

Ambient assisted living technology (AAL), also known as ambient welfare technology, seeks to use modern technology and unobtrusive systems in order to provide an extra layer of assistance to elderly people. This way, the need for a care taker can be delayed, reduced or just complemented.

AAL relies consists of a set of assistive technology. The solutions provided are based based on context aware systems, configurable human environment interaction, as well as probabilistic and rule-based reasoning, which can play an important role to solve the aging problems faced by society. It covers personalized home care assistance, smart homes, assistive cities, and related components.

An AAL space is a pervasive and transparent infrastructure that is capable of monitoring people without interfering with their life and then react to their needs and requirements. Thus, the focus of within the environment is focused around the individual. In AAL environments, the person is not expected to adapt to the technology solution, but it is the environment and technologies that have to be designed with the user in mind. The AAL environment is composed of devices and sensors, applications, services and their interfaces. Using these, the environment becomes able to sense the presence and the activities of the aging person and reacts to different situations.

6.1 Goals

The motivation of AAL can be summarized in the following points.

1. Support the aging stages at home by helping elderly people with their

activities of daily life.

2. Extend the time that elderly and disabled people are able to live independently in their preferred environment.

3. Provide personalized care and assistance with continuity, dynamically adapted to the individual needs and circumstances of the users throughout their lives.

6.2 Methods

As we mentioned, the methods to achieve those goals have to be unobtrusive, i.e., with no significant impact to the individual's life.

A context-aware system is such that is able to perceive certain aspects of the environment, like location and nearby resources. It is a component of a ubiquitous computing or pervasive computing environment. Research was being done in Xerox PARC and elsewhere on ubiquitous computing in the early 90s, from which the concept *context-aware* was born. The term was first coined in the paper [18] where the authors describe a system "in which users interact with many different mobile and stationary computers".

In general, all models of pervasive computing share a vision of small, inexpensive, robust networked processing devices, distributed at all scales throughout everyday life. A network of sensors of different types and in different locations provide information about the environment to the processing system. The sensing devices transmit the data to the processing device, which can be local or remote. A common pattern is to have a gateway controlling a few sensors and connected to other gateways or an online server that acts as the main processing unit. There might also be actuators in place that react to the current situation, in the form of, for example, intelligent light system, information shown on TV or other screens, notification over SMS or automatic emergency calls.

6.3 UbiSmart

UbiSMART, an agile framework that has the advantages of using a web platform to manage a large scale deployment; the versatility and rapidity that this solution oers in a plug and play approach; and facilitates the active

participation of end-users. UbiSMART is a generic, scalable and intuitive cloud Web-Based AAL framework.

This AAL platform converts any environment into a smart space in five minutes to assess dependent people at their homes. The final purpose of UbiSmart is to detect the ADL and provide services at the right time and through appropriate device. It minimizes the impact upon end-users by reducing the number of sensors, however, the quality of the assessment is not affected. The system is composed of three sections (Figure 6.1).

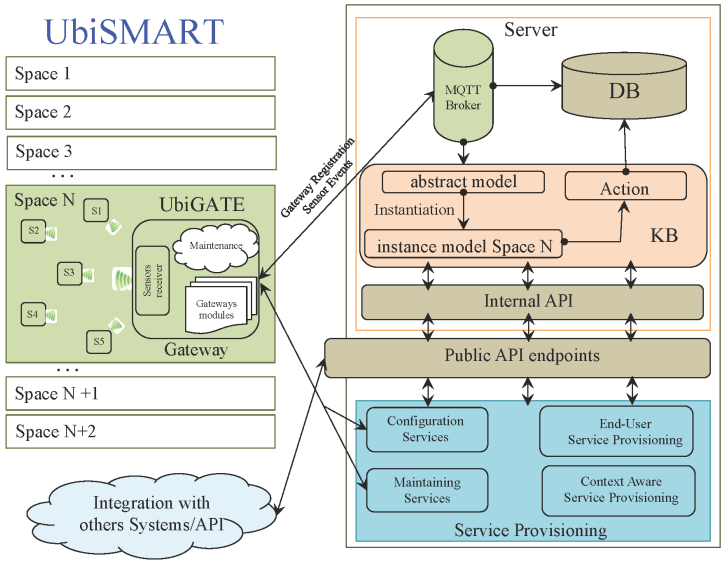

Figure 6.1: UBISmart framework in three sections: UbiGATE, the Server and the Service Provisioning [19].

First, the UbiGATE, is located in the user environment and corresponds to the Smart-home in a box package. It connects automatically to the server and the inclusion in the knowledge base (KB) is done through a dedicated service. UbiGATE is composed of multiple industrial sensors (Passive Infrared sensors (PIR) and contact sensors), an out of shell gateway (a Raspberry Pi), and a communication device (LTE or WiFi). It is in charge of processing the raw data from the sensors, converting them into events and sending them to the server via the Internet. The user receives the UbiGATE package in his environment and installs it himself by interacting with our services.

Second, the Server, which is the main brain of UbiSmart, handles the information from all the UbiGates using MQTT protocol. It employs a

semantic reasoning for the activity recognition and the service provisioning. The reasoning is based on building a generic model common for any deployment called abstract model and the specific model for each house, instance model. Together, the two models describe the entire characteristics of the smart environment. The abstract model is initialized at the UbiSmart launch and encompasses the declaration of the variables, classes, object relations, and data relations. On the other hand, the instance model is completed by the interaction of the user and it populates the KB through services.

Third, the Service Provisioning (SP), which proposes the end user a friendly web interface letting them discover the context, create knowledge and access their daily information. This section is built on a Service Oriented Approach (SOA) and it allows the user to interact with the Server without requiring any expert background, e.g., SP oers numerous services to end-users such as, a service for the automatic discovery of UbiGATE; or another service to enter specific information, such as: personal details, environment description, and sensors and device location.

6.4 Beyond the doorstep

At the start of this project UbiSmart mainly covers the indoors side of assisted living. The system is formed of motion, contact, sleep sensors and a computing unit working as a gateway; that are deployed in the place of daily living of the individuals.

However, once the person crosses the doorstep we stop having any kind of information until he or she comes back in. It's a context-aware system where the context is limited to the inside of the house.

Efforts are done in expanding the system to provide assistance related to the outdoors activities too. A change of paradigm or of expectations needs to happen, since a town or city can't be filled with sensors as easily as an individual apartment. The first part of such efforts are based on beacon technology. Beacons are low-energy devices that can be detected by other devices. They usually have a Bluetooth connection that makes them discoverable by regular smartphones. Installing some beacons in the most frequently visited places can provide a useful map where the person spends their day and how their activity changes over time. This requires of two

important things: the complicity of the neighborhood, city council, and/or certain property owners; and that the person carries a device able to detect the beacons. The second is easily solvable by using their own smartphone or providing them with one. The first one needs some work to achieve certain deals.

In addition to using ambient sensors and beacons, we can take advantage of a device carried by the users. Most modern smartphones contain a collection of sensors inside, like accelerometer and gyroscope. Their values can be used to detect what activity the carrier is engaged in. In this project we explore the activity classification based on smartphone readings as a way to expand the awareness of the UbiSmart platform.

Chapter 7

Activity Detection

We treat activity detection as a supervised learning problem. It is a classification problem, where given a certain set of smartphone sensor readings, the developed model will output a type of mobility activity: walking, biking, on a train, etc.

7.1 Sensor Types

Most smartphones with the Android operating system have built-in sensors that measure certain environmental conditions such as motion and orientation. We focus on Android because of the high availability of Android phones in the market and because the author and lab team are using Android phones.

The built-in sensors are useful to monitor three-dimensional device movement or positioning. Next we'll present the most relevant ones as seen on the Android Developer Guide [20].

7.1.1 Accelerometer

It measures in m/s^2 the acceleration force that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.

7.1.2 Gyroscope

It measures the device's rate of rotation in $rad/$s around each of the three physical axes (x, y, and z). It can be used to detect rotations (e.g. spin or turn).

7.1.3 Gravity

It measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).

7.1.4 Pressure

Acts as a barometer and measures the ambient air pressure in hPa or mbar, allowing to monitor air pressure changes.

7.1.5 Others

Other information provided by the Android Sensor API are the rotation vector, for motion detection and rotation detection; the magnetic field, that acts like a compass; and linear acceleration, which provides the acceleration force excluding the force of gravity.

7.2 Data preprocessing

We've seen that the sensors from the device will provide data in different formats and units. For example, the accelerometer gives the acceleration along the axis x (and y and z) in m/s^2, and the gyroscope provides the rotation around the axis x (and the two others) in rad/s. How do we use this data as an input to a classifier? This is what we discuss in this section.

7.2.1 Raw Data

The first option is to use the raw collected data. As we saw in the state of the art (1.2.1), accelerometer data alone has been successfully used to classify activity types. We will create a matrix were each column is a sensor and dimension (for example $accelerometer_x$), and each row corresponds to the timestamp at which all the sensor values of each column were collected.

However, since each sensor has different units and range of values, the values might differ a lot between different sensors. This can skew the models, if they end up favoring the sensors whose values reside in a bigger scale. To avoid that, we will *standardize* the data, by subtracting the mean and dividing by the standard deviation of each feature. In the resulting matrix, each column (feature) will have mean 0 and standard deviation 1.

This raw standardized data, however, won't account for the time dimension of the readings. A mobility activity has certain continuity. For example, when we walk, the conditions of our phone in a pocket change as we move our leg forward, so the readings at two different stages of the movement can be significantly different. It could also happen that one reading of a certain activity turns out to be more similar to a reading from a different activity than to one of the same activity. That is, if we look at the data as a set of static snapshots independent from each other, it might not be easy to classify them.

7.2.2 Windowed Data

To lessen the problem with the raw values, we can transform the data to treat it in a different manner. It is when we look at the progression and position of the phone over a small window of time that we can more easily tell apart the -just to give an example- circular movement of riding a bike from the step-step duality of walking.

Based on that intuition, we created a transformation of the data to group events in rolling windows of 4 seconds. The windows or frames are computed the following way: start at the first reading of the original matrix and keep adding the following readings into the first window, until we reach a row that was recorded ≤ 2 seconds after the first row. Then a second window is started, and the process is repeated until all data is processed and sorted into windows. From the values contained in each window, we then take statistical measures such as the mean and median. Note that the number of data points contained in each window is not fixed.

In the resulting matrix, each row represents one such window and each column contains a measure of the events grouped in the window. Thus, some example of features will be: *mean_accelerometer_x*, *median_accelerometer_x*, *max_gyroscope_z*, *percentile_75_magnetic_field*, etc.

7.3 Choice of Algorithms

In this section we present some classic machine learning methods that are popular and/or have been shown to provide certain results when used to classify mobility activities.

7.3.1 Naive Bayes

Bayes' theorem, also know as Bayes' rule, describes the probability that an event takes place based on prior known conditions. Let A and B be events, and $P(B) \neq 0$, then the Bayes' theorem is stated as:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

Naming y the output class and considering the input feature vector x_1 through x_n, the previous relationship can be expressed as the following:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

A classifier using Naive Bayes methods assumes independence between any pair of features, i.e., that the presence of a particular feature in a class doesn't affect the presence of any other feature. Using this naive independence assumption,

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$$

which can be simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

for all i.

This assumption makes Naive Bayes classifiers significantly faster than more sophisticated methods, since each class distribution can be independently estimated as a one dimensional distribution; while also requiring less training data.

Even though their naive assumption, which we can't say that matches our data, Naive Bayes models have been shown to be useful for many classification tasks, including spam filtering and activity detection.

7.3.2 k-Nearest Neighbors

Nearest Neighbor (k-NN) methods stem from the idea of classifying observations based on the labels of their nearest neighbors. When used to classify an element, the k closest training samples are used to determine the output

class of the element. It is chosen by a majority vote of its neighbors, with the object being assigned to the class that is more common among its k nearest neighbours. $k > 0$, and it is an integer.

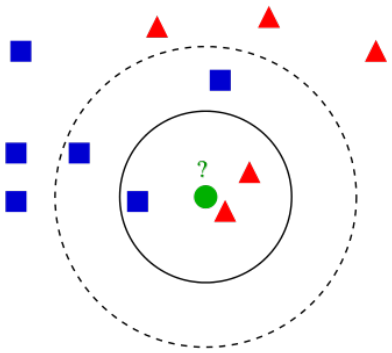

Figure 7.1: Example of k-NN classification (source: Wikipedia)

Training the classifier consists simply of storing the feature vectors of the training data and their class label. To classify an unlabeled vector, a k is chosen by the user and the vector is classified by giving it the label that has more occurrences among the k training samples that are closest to the given vector. In Figure 7.1, we can see a visual example. The green circle is the element we want to classify. If we choose $k = 1$ or $k = 2$, it will be assigned the class of the red triangles, which are the nearest and second nearest samples. It will also be classified a as a red triangle if we choose $k = 3$, since 2 out of the 3 nearest training samples are red triangles. If $k = 5$, the majority of elements among the 5 closest ones (inside the dashed line circle) are blue squares, so that will be the assigned class this time.

7.3.3 Support Vector Machines

Support Vector Machines (SVM) are supervised learning methods used for classification, regression and outlier detection. They work well in high dimensional spaces, are versatile (different kernel functions can be specified), and are usually memory efficient.

Imagine we have a p-dimensional vector representing a data point that we want to classify. If we have two possible classes, a SVM attempts to find a way to separate their points with a $(p-1)$-dimensional vector. A good hyperplane that separates the data could be one leaving the largest margin between the two classes. The training process consists of looking for such a

plane. In the classifying phase, samples will be assigned a class depending on which side of the hyperplane they are located.

To do multi-class classification with a Support Vector Machine, we can use "one-against-one" or "one-vs-rest" approaches. The former trains as many classifiers as pairs of classes: $n\_class * (n\_class - 1)/2$ if n_class is the number of classes. One-vs-rest, on the other hand, trains n_class models.

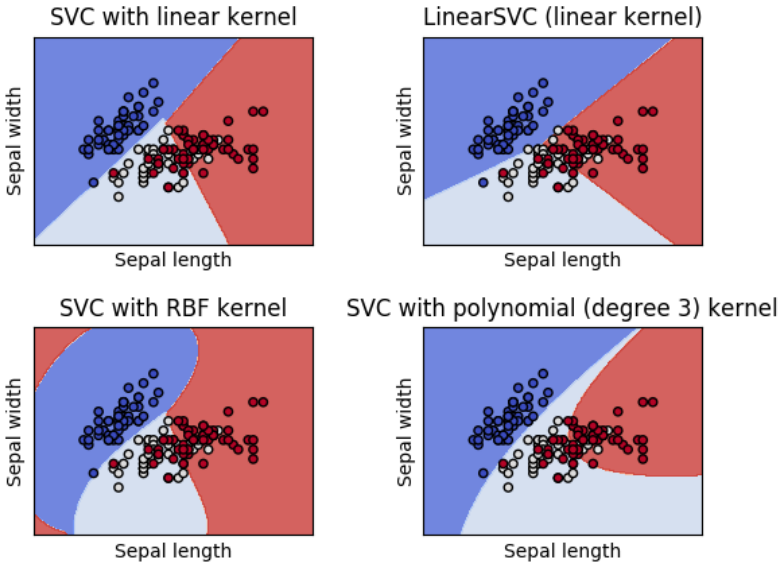

Figure 7.2: Different kernels yield different models (source: sklearn documentation)

7.3.4 Discriminant Analysis

Two other classic classifiers are Linear Discriminant Analysis and Quadratic Discriminant Analysis. Their decision surface is linear and quadratic, respectively.

Both LDA and QDA model the class conditional distribution of the data $P(X|y = k)$ for each class k through simple probabilistic models. Then Bayes' rule can be used to predict, by computing the probability for each class and selecting the class k that maximizes it. For both LDA and QDA, $P(X|y)$ is modeled as a Gaussian distribution with density:

$$p(X|y=k) = \frac{1}{(2\pi)^n|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t\Sigma_k^{-1}(X-\mu_k)\right)$$

For classification, the training data has to be used to estimate the class priors $P(y=k)$, the class means μ_k and the covariance matrices.

Chapter 8

Dataset Collection

As we introduced before, we are dealing with a supervised learning problem. To train the classifiers we need a labeled dataset. We decided to collect our own, this section describes the method used for that purpose.

8.1 Dataset Design

We need a dataset that contains the following activity labels: walk, run, bike, train, car, bus. The readings should come from a smartphone worn in a pocket, and include the multiple sensors described in 7.1. There are human activity datasets available online [21, 22]. However, no dataset fulfilling those condition was found, so we decided to collect our own. For this reason, we developed an Android application that would let us collect a labeled dataset.

8.2 Activitrack Design

We developed an Android app and named it *Activitrack*. In this section we will explain its design and implementation.

8.2.1 Target users

It will be used by the author and people close to our researchers, in natural conditions when going on with their life. Our app has to accommodate that. Note that this app is not intended for the final users (elderly people), but for volunteers that decide to label their activities and help us build a dataset of sensor readings.

8.2.2 Interface

The interface should be simple. The use flow of the app will be:

1. Open application (when the user decides to record an activity).
2. Tap *Record*.
3. Choose an activity from the list.
4. Carry out that activity while carrying their phone.
5. Tap *Stop*.
6. Upload the generated datasets. This does not need to happen every-time. When the user taps *Upload*, the CSV files recorded so far will be sent to a server run by the author.

Figures 8.1 to 8.4 show the user interface of the app while going through this process.

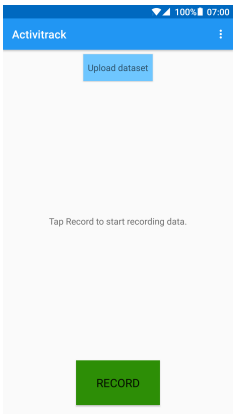

Figure 8.1: Activitrack main view

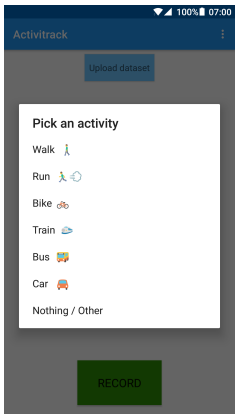

Figure 8.2: View to add a label

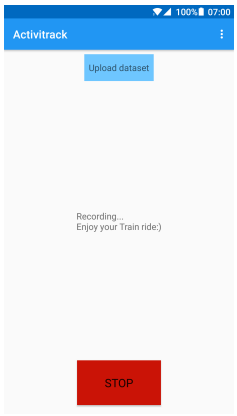

Figure 8.3: View to

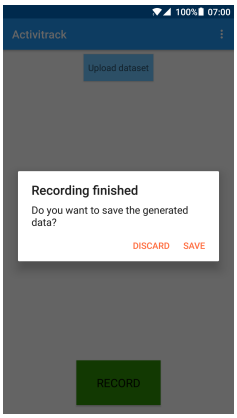

Figure 8.4: A really Awesome Image

8.2.3 Challenges in sensor detection

The dataset that we want to generate has the shape of a table with different time instants as rows, and the value of each sensor at that moment as columns. The way to access the device sensors from an Android app is

through the Sensor Manager API. It does not allow to request the value read by a certain sensor at one particular time, which would make it very direct to generate the wanted table. Instead, a Sensor Manager class has to be used to *subscribe* to a sensor or collection of sensors. A listener function then is implemented to process each event that the manager collects.

In more detail, one subscribes to a particular sensor choosing a certain frequency and defines what function will be called when there's a new value available. Then, that function is called by the system with the specified frequency to report the value read by the sensor at that time. This allows the application to read from one or multiple sensors with the desired frequency. However, even if the frequency chosen for different sensors is the same, the subscription to events can't be done for more than one sensor at a time, and there will be a delay between readings received. In fact, there is another challenge: the frequency of a sensor subscription is only a *suggestion* and the real frequency through which the Android API provides events is unknown and variable.

On another side, we have to consider the database writing speed. We will choose a small frequency to the sensor subscriptions – we want to read sensors every few hundredths of a second (300-500 ms) and the delay between firing of two events of different sensors might be much smaller than that. When reading at high speeds, trying to write every sensor reading directly to the database could be problematic. Accessing the disk can be a slow operation and trying to do successive writes repeatedly could produce conflicts and it is not necessary. To cope with this and the previous problems, we decided to keep the latest few readings in memory and then store them at once to the SQLite database.

8.2.4 Database design

As we have already mentioned, we decided to create one single database table where each column is a sensor reading or information about the raw (e.g. the timestamp). Another option we considered was having multiple tables, one per sensor or piece of information. This way each sensor value would have its own exact timestamp. The drawback is that we would have to do some more work when preprocessing the data, grouping together the events that happened within a close window of time. Therefore, we decide to do this grouping from the Android app and create one single table.

id	timestamp	label	android.sensor.gyroscope	android.sensor.gyroscope_uncalibrated	android.sensor.gravity	android.sensor.accelerometer
1	600905424629104	Train	1.0863495;-1.8414459;0.1831	1.0688019;-1.8365631;0.158432;-0.01754	3.5826766;6.7727427;6.1	0.38082886;5.5072174;3.966354
2	600906889935353	Train	0.39979553;0.85401917;0.07	0.38224792;0.858902;0.04777527;-0.017	8.014274;0.3681325;-5.63	8.514145;-1.1074371;-5.8320312
3	600908457311498	Train	-0.18579102;0.061706543;0.5	-0.20333862;0.066589355;0.52742004;-0	0.45171502;-1.5886751;-9	0.8310089;-0.70762634;-10.5560
4	600910041576706	Train	0.025177002;0.011810303;0.0	0.0076293945;0.016693115;0.038619995	-0.35068512;-2.6970735;-	-0.9455414;-2.079712;-9.821747
5	600911625176810	Train	0.009399414;0.0019226074;0	-0.008148193;0.00680542;0.018829346;-0	-0.75255686;-2.0944583;-	-0.82691956;-2.106903;-9.86676
6	600913500353163	Train	-6.713867E-4;0.016098022;0.	-0.018218994;0.020980835;0.02961731;-0	-0.76182854;-2.0661943;-	-1.0008392;-1.9530334;-9.40129
7	600914775727902	Train	0.016815186;0.062149048;0.0	-7.324219E-4;0.06703186;0.027267456;-0	-0.8762091;-1.9301376;-9	-1.0829468;-1.7127991;-9.45756
8	600916357437172	Train	-0.06428528;-0.07551575;0.0	-0.081832886;-0.070632935;0.004425049	-0.98127645;-1.8014227;-	-0.74783325;-2.0449219;-10.144
9	600917942272120	Train	0.002670288;-6.2561035E-4;0	-0.014877319;0.004257202;-0.012435913	-0.8236301;-2.066925;-9.5	-0.7403412;-2.1520844;-9.69651
10	600919527843057	Train	-0.0046539307;0.0063476562	-0.022201538;0.011230469;-0.019760132	-0.76879245;-2.299211;-9	-0.6347809;-2.3484497;-9.74089
11	600921091094045	Train	-0.0030975342;-6.5612793E-4	-0.020645142;0.0042266846;-0.02224731	-0.80233437;-2.2771854;-	-0.77986145;-2.3019714;-9.7538
12	600922674906128	Train	-0.007461548;-0.0031280518	-0.025009155;0.0017547607;-0.02488708	-0.78935736;-2.295334;-9	-0.7658844;-2.1421967;-9.71218

Figure 8.5: Sample of data recorded with activitrack (multiple columns missing)

Table 8.5 shows a few sample rows of data collected with Activitrack.

8.3 Collection Method

Once the application was ready, a few users "wore" the app while doing some activities in order to provide us with data. We also developed an HTTP server to which the data can be easily uploaded from the app. It was done in Python with the Flask server framework. Its task is rather simple: collect any CSV files it receives and store them under a directory for the corresponding user.

We had 5 users, including the author and the supervisor, track some of their daily activities while using the app. We obtained a total of more than 55000 data points (rows).

Chapter 9

Experiment and Results

9.1 Experiment design

The structure of the experiment is the following.

1. Preprocess data.
2. Set aside 20% of the data for validation.
3. Make cross-validation splits.
4. Train and test each split with the different classifiers.
5. Choose the best model(s) according to their average performance across splits.
6. Use the validation set to obtain the general performance of the chosen model(s).

It is important to create a validation set to test the final models. This lets us see how a model performs with previously unseen data, and gives us an idea of how well it would generalize. If we didn't do this, and used the same data used for cross-validation to later test the chosen model, we would have a biased idea of its performance, since the best model was chosen based on its performance on parts of that same set. Thus, after preprocessing we take 20% of the data and put it aside to test the final models. We do it after shuffling the samples, and preserving the percentage of samples of each class. The validation data is set aside and from now on until the end of this

section, when we talk about the data we mean the remaining 80% unless otherwise specified.

To compare the different models we use cross-validation. In particular, we use k-fold cross validation, with $k = 4$. Our dataset is not small but it is not extremely big either. Choosing a much higher number of folds would mean that each model would be tested with fewer data in the splits. We choose 4 so that at each fold 25% of the model data is reserved for testing. That means that we split the data (the 80% remaining after we take out the validation set) in 4 parts, yielding 4 splits of train-test sets. That is, each split uses 75% of the data to train the model and we test it on the other 25%. The folds are selected after shuffling the dataset, and are made by preserving the percentage of samples for each class (known as stratified folds). As a scoring method, that is, the way to test the models and compare them, we use the average accuracy. It is the percentage of test samples classified correctly, averaged for the 4 folds.

The models that we test are: Gaussian Naive Bayes, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Support Vector Machines with RBF kernel, and SVM with polynomial kernels with degrees 2 to 5. For each split, we fit each classifier to the training data and test them on the test data. This gives us the performance of each model for each fold. We compare their accuracy and decide on what model to keep.

The chosen model is then trained with the data that comprises all folds, and we use the other 20%, that we set aside earlier, to validate it and obtain an estimation of its general accuracy, the recall and precision, and F-measure.

The same procedure is used for the windowed data. Its preprocessing step includes generating a matrix where each row is a window and the columns contain the stats for the window. Then we do cross-validation to compare the different models.

9.2 Results

9.2.1 Raw data

Table 9.1 shows the results using the raw data. The mean accuracy is the average between the accuracy of each fold, each of them computed as the percentage of test samples classified correctly. We also show the standard

deviation – the accuracy differs for each fold and it is important to know *how much*. The third row has the execution time in seconds to get an idea of how computationally expensive it is to train and test each classifier. It is the total time for each model to train and test on the 4 folds.

Model	**Mean accuracy (%)**	**std**	**Execution time (s)**
GNB	64.86	0.39	0.57
LDA	59.11	0.23	2.11
QDA	71.15	0.47	2.61
3-Nearest Neighbors	96.06	0.07	12.63
4-Nearest Neighbors	95.60	0.10	13.69
5-Nearest Neighbors	95.58	0.08	14.54
SVC poly degree 2	84.39	0.14	115.46
SVC poly degree 3	85.71	0.07	122.11
SVC poly degree 4	84.23	0.12	117.25
SVC poly degree 5	81.29	0.35	138.42
SVC rbf	89.34	0.19	166.09

Table 9.1: Raw Data: cross-validation scores

We see that Naive Bayes and Linear Discriminant Analysis are really fast but don't perform very well. Quadratic Discriminant Analysis achieves a better accuracy but still 10 points below than other classifiers. Support Vector Classifiers achieve pretty good results, between 84% and 89% mean accuracy. With the polynomial kernels we see that the accuracy decreases after degree 3, maybe due to some overfitting to the training data. The highest accuracy is achieved by the radial basis function kernel, (89.34% and std=0.19). It comes with a cost, having the longest execution time: 166 seconds. The Nearest Neighbors classifiers take ten times less to run – and they have a better accuracy. The chosen N does not make their accuracy change a lot, but having $N = 3$ seems to yield a slightly higher accuracy, with the lowest variance and lowest execution time (by a small margin). For this reason, we choose the 3-Nearest Neighbors model as the best classifier for our raw data.

As we saw in Section 9.1, the last step is to train this model with the whole data we used for cross-validation, and test it with the remaining 20% that we out aside previously. **The accuracy achieved is 96.20%**.

Next, we explore more details about the classification done on the validation set. We show a classification report in Table 9.2. It includes precision, recall, f1-score and support. To explain their meaning, let TP, FP, TN, FN mean True Positives, False Positives, True Negatives, and False Negatives, respectively.

- **Precision** It is computed as $\frac{TP}{TP+FP}$. An intuitive example of what it means would be "Out of all the samples that were classified as Bike, how many (what ratio) were in fact Bike events?". A precision of 1.0, we can see on *Nothing*, means that 100% of the samples it classified as *Nothing* were indeed *Nothing* events. Precision is pretty high across the different classes. The class with lowest precision is *Bus*, which means that we have some events classified as *Bus* when they are actually something else. A hypothesis to explain that would be that they are classified as *Car*, a vehicle of the same nature.

- **Recall** It is computed as $\frac{TP}{TP+FN}$. For example, "Out of all the Bike events, how many did were actually classified as Bike?". The recall is higher than 90% for all classes, *Bus* being again the lowest with 91%.

- **F1-score** It is computed as $F1 = 2 * \frac{(precision*recall)}{precision+recall}$, and it can be interpreted as a weighted average of the precision and recall.

- **Support** This is simply the number of samples of each class in the validation set.

The averages of the last row are weighted by support.

To test the idea that some bus events might have been classified as car, we did a confusion matrix. A confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j. We check the row of bus samples and we see that 98.9% were classified as *Car* (correctly), but some were classified as Walk, Train or Car: 2.51%, 3.01% and 3.85% of them, respectively. We see that it is true that a slight confusion between car and bus takes place, but we can consider it normal. It is interesting to see that some bus samples *look like* walk or train events to the model. Our intuition tells us that situations like standing on a bus, train or at a red light when walking are not so different from each other.

class	precision	recall	f1-score	support
Bike	0.97	0.96	0.96	1319
Bus	0.87	0.90	0.89	598
Car	0.97	0.99	0.98	3644
Nothing	1.00	0.99	0.99	547
Train	0.93	0.91	0.92	1227
Walk	0.97	0.96	0.96	3773
avg / total	0.96	0.96	0.96	11108

Table 9.2: Classification report. 3-nearest neighbors model, with the raw data

First, we plot the data in a 2D projection over the first two Principal Components. In Figure 9.1 we can get an idea of how the data looks. We can see that some classes are very well separated from the others. However, there are also some areas shared by many labels, where it is more difficult to imagine a border. We cannot infer a lot from just one plot, but our intuition tells us that we should be able to find a model that fits the data well enough.

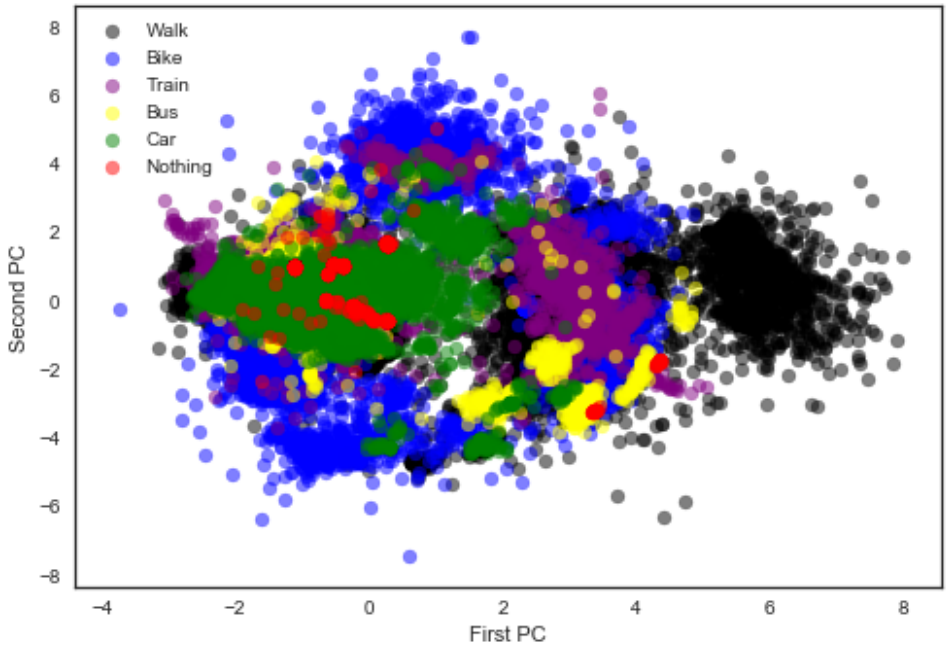

Figure 9.1: Model data (80% of the total collection) projected over its two first principal components

9.2.2 Windowed data

We explained in Section 9.1 how we create a windowed dataset. Using rolling windows of 2 seconds, we build a matrix X_w with shape $(7240, 136)$. Obviously, now we may have many less rows than with the raw data, but more columns due to the multiple measures considered for each sensor. The stats added to X_w for each sensor dimension are: 25th, 50th and 75th percentiles, mean, median, standard deviation, minimum, and maximum.

Using the windowed data, the cross-validation process to compare the different models gives the results shown in Table 9.3. We notice that a general increase in accuracy happens across most models. Naive Bayes, which delivered less than 65% of accuracy with the raw data, gives a mean accuracy of 85% here. This makes us think that, indeed, grouping close events together makes it easier to distinguish different activities. As a deployable model we choose any of the k-Nearest Neighbors, which give $> 98\%$ accuracy and are fast to train.

Model	**Mean accuracy (%)**	**std**	**Execution time (s)**
GNB	85.08	1.03	0.22
LDA	90.26	0.32	2.94
QDA	91.75	2.60	4.82
3-Nearest Neighbors	98.86	0.14	1.62
4-Nearest Neighbors	98.46	0.16	1.55
5-Nearest Neighbors	98.14	0.11	1.49
SVC poly degree 2	98.83	0.14	7.71
SVC poly degree 3	98.22	0.13	4.45
SVC poly degree 4	98.93	0.18	4.10
SVC poly degree 5	98.52	0.10	4.57
SVC rbf	88.92	0.20	43.81

Table 9.3: Windowed Data: cross-validation scores

Let's take the 3-Nearest Neighbors classifier and compute the validation accuracy. We train it with the whole model data and then test it with the 20% of the data we initially set aside. **The accuracy achieved is 96.48%**.

Once more, we compute the precision, recall and F1-score for each class and present it in Table 9.4. Note that the support is much lower: the dataset from creating the windowed groups has 5792 rows for modelling, and 1448

class	**precision**	**recall**	**f1-score**	**support**
Bike	0.94	0.99	0.97	103
Bus	0.97	0.99	0.98	398
Car	0.96	1.00	0.98	96
Nothing	0.99	0.99	0.99	174
Train	0.95	0.95	0.95	480
Walk	0.96	0.90	0.93	197
avg / total	0.96	0.96	0.96	1448

Table 9.4: Classification report. 3-nearest neighbors model, with the raw data

for validation. The three measures shown are high, especially the recall.

Chapter 10

Model Deployment

In the previous chapter we developed a model that can classify sensor data into the activity type. The code was developed on a Jupyter notebook and Python scripts. How do we turn those into an deployed service that can be used in the environment of UbiSmart? In this chapter we answer this question.

The final integration with the system will consist of:

1. An Android application that periodically records sensors values and sends them to...
2. An HTTP server that listens to data from the Android app.
3. A Python class that wraps the methods around the model and predicts the user activity.
4. A controller in the existing UbiSmart server (see Section 11.2).

To create this new piece of the system and later integrate it with UbiSmart, we choose to follow a microservice architecture.

10.1 A classifier service

10.1.1 Microservice architecture

A microservice architecture is a type of software architecture in which an application is structured as a collection of loosely coupled services. As Martin Fowler describes them [23], they differ from libraries (another way of

componentizing software) in that services are not called using in-memory function calls, but rather communicate with mechanisms such as web service requests.

Services are independently deployable, and this is one main reason for using services as components instead of libraries. In an application based on services, a change in one of them requires only to redeploy the changed service. If the components are made of libraries, a change in one of them will result in having to deploy the whole project again.

In a microservice architecture, the services are organized in a business capability. That is, one service will tackle a business problem or a set of features of a product, and include all the layers required to make it a standalone problem (like user interface, persistent storage, etc.).

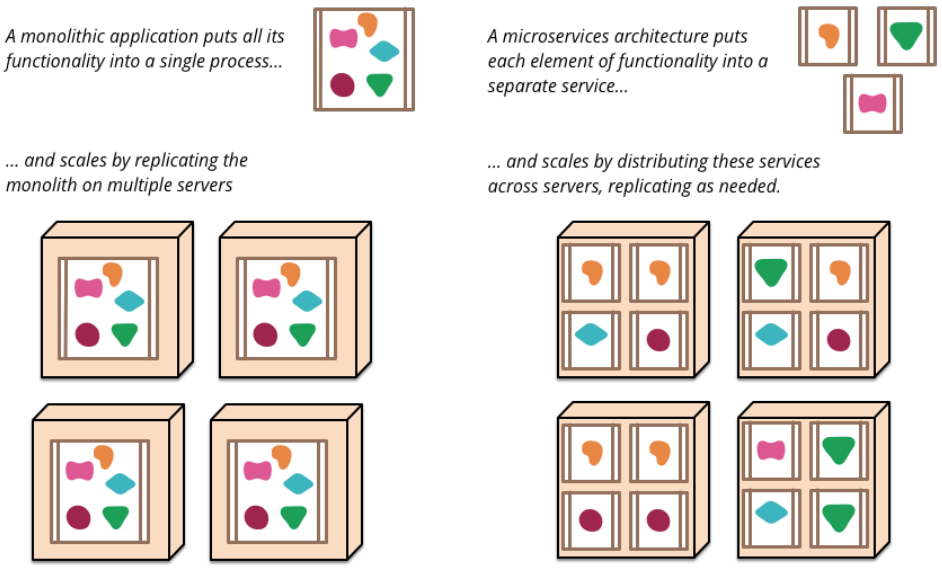

Figure 10.1: Monoliths and microservices [23].

There is discussion around what should be the size of a microservice in order to actually be *micro*. We won't entertain in the debate and when applied to our case we will use the words *service* and *microservice* interchangeably.

The current system, as portrayed in Section 6.3, has a monolithic architecture. The web server is programmed as a single piece. For different test sites with elderly people, different instances are deployed and run indepen-

dently from each other.

In an effort to avoid the main server from growing too much, we decided to prepare our classifiers as part of a service. This also makes it deployable by its own, and it makes it easy to use a different programming language. The UbiSmart web server is in Node.js, we use Python 3 for our service.

10.1.2 Classifier server

The server uses Flask, a microframework for Python to make web servers. The role of our server is to listen for POST requests at a certain endpoint and process the data that it receives. This data is a JSON or CSV sent from a user's smartphone with one or multiple sensor readings. To process the data, it uses a class we made, `ModelWrapper`, which manages the model methods. The server will call its `predict` function passing the received data. This way we get the activity prediction for each row. In case the loaded model is based on the rolling windows, the number of predicted activities can be lower than the number of rows received.

The following code snippet shows the main logic behind the endpoint that receives the data files. We should note that `mw` is an instance of the `ModelWrapper` class, and the method `send_result(res)` connects to the UbiSmart web API and sends the list of predicted classes.

```
@app.route("/predict_activity", methods=["POST"])
def predict_activity():
    if 'csv_file' in request.files:
        received_file = request.files['csv_file']
        print("File %s received. Reading and predicting..." %
            received_file.filename)
        try:
            res = list(mw.predict_csv(received_file))
            send_result(res)
    ## ...
return jsonify(res)
```

As for the `ModelWrapper`, its job is to manage and use an activity classifier. It contains methods to load a stored model, train it, and test data using the model. It also includes the preprocessing required. All the calls provide parameters to control the desired behavior.

10.2 A client application for end-users

After the classifier is online, how do the end-users (elderly people) make use of it? The use that our system will have needs to be passive. The user is not expected to learn how to use an application that sends data. Instead, such an application is to be transparent. We developed a simple app to be able to have an end-to-end demo of the system. The application does the following: on start, it uses Android's Alarm API to create a new (silent) recurrent alarm. Instead of using the Alarm Manager class, we first intented to use the Job Scheduler. Every 5 minutes approximately, the alarm will fire. We subscribe to all the sensors and read their reported values for a few seconds. Then, we unsubscribe and turn the sensors off. The recorded data is stored persistently on an SQLite database. Finally, if an internet connection is available, the stored data will be transmitted to the server and eventually removed from the phone. If that isn't possible, the data will be transmitted (or attempted to) on the following alarm firing.

Figure 10.2 shows the resulting architecture after adding the classifier microservice. In the image, the phone gateway is named UbiTracks, a temporary name given to the user application explained in this section.

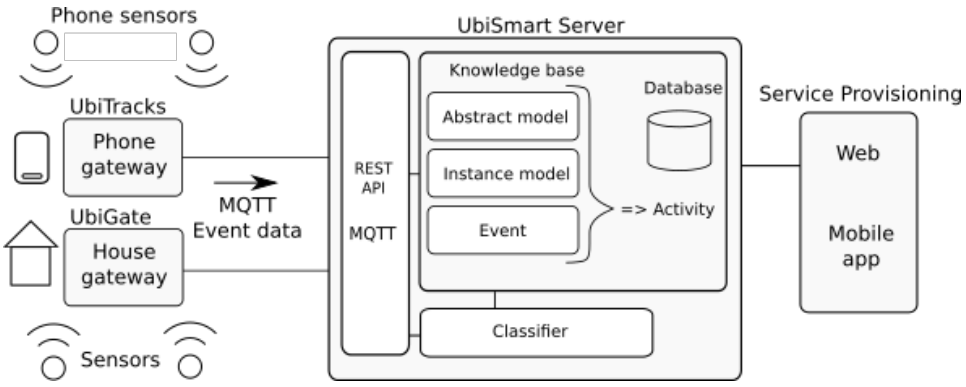

Figure 10.2: System architecture after adding classifier system

Chapter 11

Ontology

The UbiSmart platform contains an ontology and knowledge-base that power the reasoning behind the user needs. In this chapter we explore its details and expand it to integrate the knowledge provided by the activity classifier.

11.1 Knowledge-Based Systems

Knowledge-based systems (KBS) are computer programs that reason and use a knowledge base to solve complex problems. They attempt to represent the world or its information not implicitly through code but explicitly using ontologies or similar tools. An ontology is a formal way of describing the entities that exist for a particular domain, defining the types, properties and interrelations between entities. Both *ontology* and *KBS* are terms used frequently on classic artificial intelligence, field where they were coined. A knowledge base represents facts about the world, and the system includes an inference engine that applies logical rules to the knowledge base to deduce new information.

The **Resource Description Framework** (RDF) is a family of specifications by the World Wide Web Consortium (W3C) which provides a general method for conceptual description or modeling of information that is implemented in web resources. It was originally designed as a metadata data model. It is part of the **Semantic Web**, an extension of the World Wide Web that has the goal of providing a common framework for data to be shared and reused across application, enterprise, and community boundaries.

RDF uses subject-predicate-object expressions, known as triples, to make statements about resources (usually web resources). As an example, we can represent the notion "The patient is at home" in RDF as the triple: a subject denoting "the patient", a predicate denoting "is located at", and an object denoting "home".

The knowledge of UbiSmart is described in Notation3, N3 in short, which is a serialization of RDF models. It is designed with the focus on human readability, and it is more compact and readable than XML RDF. Figure 11.1 shows some classes and properties to model the frailty of a person.

```
## Workspace definition ##
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
@prefix qol: <http://www.ubismart.org/n3/qol-model#>.
## CLASSES ##
qol:User a rdfs:Class.
qol:Service a rdfs:Class.
qol:Frailty a rdfs:Class.
qol:frailtyItem a rdfs:Class;
    rdfs:subClassOf qol:Frailty.
qol:Device a rdfs:Class.
## OBJECT PROPERTIES ##
qol:hasFrailty a rdf:ObjectProperty;
    rdfs:domain qol:User;
    rdfs:range qol:Frailty.
qol:hasService a rdf:ObjectProperty;
    rdfs:domain qol:frailtyItem;
    rdfs:range qol:Service.
## DATATYPE PROPERTIES ##
qol:itemValue a rdf:DatatypeProperty;
    rdfs:domain qol:frailtyItem;
    rdfs:range xsd:integer.
qol:frailtyDetected a rdf:DatatypeProperty;
    rdfs:domain qol:frailtyItem;
    rdfs:range xsd:boolean.
```

Figure 11.1: Modelling human frailty with Notation3 [12].

11.2 Ontology expansion to outdoors

In Figure 11.1 we saw part of the N3 code that represents the frailty of users. That is part of the ontology that represents the knowledge map of UbiSmart. The interface of the system with the users consists of the collection of sensors and gateway deployed at their living space, plus a web site

where the data is graphically presented. The information detected by the sensors, and collected by the gateway, is translated to triples and then processed by the server. It is treated as knowledge and added to the knowledge database. The inference program runs periodically and this addition might be used to infer new knowledge about the user or their environment. In addition, the rules inputs and outputs can contain both certain and uncertain information. To show an example, Figure 11.2 shows a semantic rule without considering uncertainty, and the same rule with uncertainty added.

Classical representation

$\forall$ *Sensor se; SensorState st; Location l; Resident r*
(se hasCurrentState ***st****))*
$\wedge$ *(se deployedIn l)* $\wedge$ *(r liveIn l)*
$\Rightarrow$ *(r detectedIn* ***l****)*

Uncertainty inclusion

$\forall$ *Sensor se; SensorState st; Location l; Resident r*
(se hasCurrentState ***[a Uncertainty; certaintyLevel n; relatedObject st])***
$\wedge$ *(se deployedIn l)* $\wedge$ *(r liveIn l)*
$\Rightarrow$ *(r detectedIn* ***[a Uncertainty; certaintyLevel n; relatedObject l; accordingTo se])***

Figure 11.2: Rules for detection of sensors with and without uncertainty [11].

Each house, or end-user, has a knowledge base associated. This is created at deployment time, based on the general knowledge or ontology, and the details of their deployment. Figure 11.3 shows part of the ontology for one patient, detailed around the Sensor class.

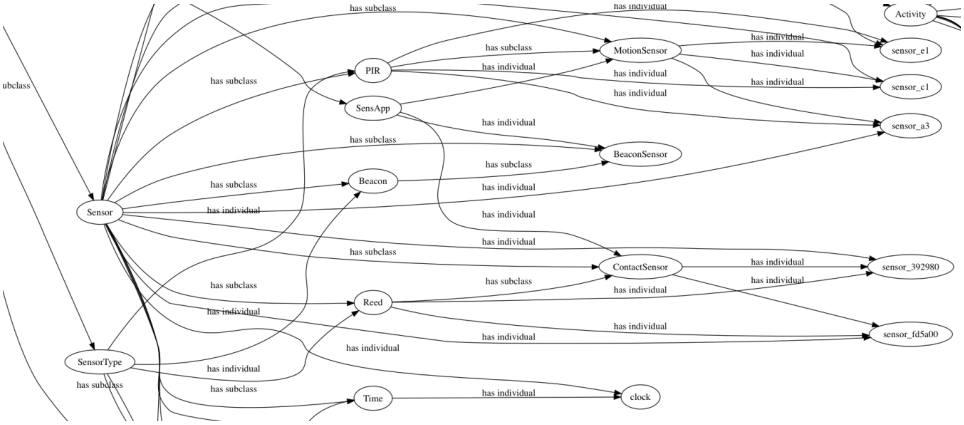

Figure 11.3: Sensor-related classes and instances in the ontology

Current sensors include motion sensors (Passive Infrared Sensor, PIR), contact sensor, and mattress sleep sensors. We need the ontology to include information about our mobility classifier. It makes sense to treat our deployed classifier as a sensor of its own: it is deployed on its own, and communicates through HTTP with UbiSmart to send the detected activities.

The part of UbiSmart that manages the web communication of the gateways is managed by a node.js web server. We made our classifier microservice to send the detected information to UbiSmart, so we added an endpoint to UbiSmart and created a new controller that receives and processes this information. To allow the reasoning motor of UbiSmart to represent this new sensor, we only need to add the following to each house knowledge base.

```
hom:mobilitySensor a qol:Sensor;
       qol:id "mobility"^^xsd:string;
       rdfs:label "Mobility sensor through a smartphone provided activity recognition"@en.
```

Figure 11.4: Snippet to add to the knowledge base to support our mobility *sensor*

Chapter 12

Conclusions and Future Work

We started this project with the goal of expanding an Ambient Assisted Living platform to include outdoor capabilities, specifically by adding mobility type detection through smartphone sensors. The main objective, then, was to create a classifier that accurately distinguishes between different activities (walk, bus, bike,...). We expected it to be possible because very similar problems have been solved in the literature, but we faced additional challenges like the lack of a dataset generated in controlled conditions.

We overcame that by developing an Android application that makes it easy to label activity data and send it to our server. This way we collected a dataset of more than 55000 samples. We explored two different ways of processing the data, one based on the original samples and the other based on a rolling windows transformation. After the data processing and analysis, we can say that we achieved the main objective: building a model able of classifying mobility activities with high accuracy according to the testing done.

The model in isolation would not fulfill the main goal of expanding the existing AAL platform, so a lot of work on integration was also done. We developed an independently deployable microservice that delivers the activity detection as a web service. It sends the detected activity to UbiSmart, where we added a controller that handles it and turns it into treatable knowledge. After research on knowledge-based systems, the existing ontology was briefly extended to include knowledge about mobility. A new secondary goal

was added after the initial plan – to develop an Android application that automatically uses the sensors to periodically track and send their readings to the deployed server. This was achieved successfully and concludes an end-to-end system that demonstrates the use cases of this project.

12.0.1 Future Work

The first way to improve the work done would be to collect a bigger, broader, more representative dataset. Our model achieved good results and we validated it with rigor, but we can't expect it to have the same accuracy for all new users or phones used. That's why it would be worth it to spend some resources on having a variety of users label their daily mobility for a period of time. The code written was developed with that in mind, and the classifier microservice contains methods for retraining, loading and storing models to disk – to minimize the changes needed in the future.

Another direction to expand our project upon is the ontology. It can be studied for further expansion, for example by relating the mobility types (now considered as *sensor events*) to *activities*. Depending on the needs of the caregivers and users, some rules can be added to the inference base to increase the information obtained about the elderly's status and health.

12.0.2 Impact

The work presented here paves the way for IPAL's system of AAL to *cross the doorstep*. The indoor system has been deployed in a few tens of houses in Singapore and France with positive feedback. Adding the dimension of outdoor mobility provides a direct improvement to the platform, and can help the users' family and caregivers provide a better assistance to the elderly. We are satisfied to have this positive social impact.

iiWAS 2017 (Salzburg, Austria)

The work of this thesis has been summarized and included in a paper by Martin Kodyš, Pau Oliver, Joaquim Bellmunt and Mounir Mokhtari. With the title *Modeling Human Mobility in Urban Living Environments*, it has been accepted for presentation at the 19th International Conference on Information Integration and Web-based Applications & Services (iiWAS2017).

Bibliography

[1] United Nations Publications, ed., *World Population Ageing 2015: Highlights*. New York: United Nations, 2016.

[2] M. Mokhtari, R. Endelin, H. Aloulou, and T. Tiberghien, "Measuring the impact of ICTs on the quality of life of ageing people with mild dementia," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence a2nd Lecture Notes in Bioinformatics)*, vol. 8456, pp. 103–109, 2015.

[3] "Active and Assisted Living Programme: Objectives." http://www.aal-europe.eu/about/objectives/, 2017.

[4] M. Mokhtari, H. Aloulou, T. Tiberghien, J. Biswas, D. Racoceanu, and P. Yap, "New Trends to Support Independence in Persons with Mild Dementia – A Mini-Review," *Gerontology*, vol. 58, no. 6, pp. 554–563, 2012.

[5] E. Munguia Tapia, S. S. Intille, and K. Larson, "Activity Recognition in the Home Using Simple and Ubiquitous Sensors," *Pervasive Computing*, vol. 3001, pp. 158–175, 2004.

[6] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The Gator tech smart house: A programmable pervasive space," *Computer*, vol. 38, no. 3, pp. 50–60, 2005.

[7] A. Marinc, C. Stocklöw, A. Braun, C. Limberger, C. Hofmann, and A. Kuijper, "Interactive personalization of ambient assisted living environments," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6771 LNCS, pp. 567–576, 2011.

[8] A. Morales, T. Robles, R. Alcarria, and D. Alonso, "Communication Architecture for Tracking and Interoperable Services at Hospitals: A Real Deployment Experience," in *Lecture Notes in Computer Science, Ambient Assisted Living Third International Workshop*, pp. 84–91, Springer Berlin Heidelberg, 2011.

[9] J. Cubo, A. Nieto, and E. Pimentel, "A Cloud-Based Internet of Things Platform for Ambient Assisted Living," *Sensors*, vol. 14, pp. 14070–14105, aug 2014.

[10] J. Bellmunt, T. Tiberghien, M. Mokhtari, H. Aloulou, and R. Endelin, "Technical Challenges Towards an AAL Large Scale Deployment," in *Lecture Notes in Computer Science, International Conference On Smart homes and health Telematics*, pp. 3–14, 2015.

[11] H. Aloulou, M. Mokhtari, T. Tiberghien, R. Endelin, and J. Biswas, "Uncertainty handling in semantic reasoning for accurate context understanding," *Knowledge-Based Systems*, vol. 77, pp. 16–28, 2015.

[12] J. Bellmunt, M. Mokhtari, B. Abdulzarak, H. Aloulou, and M. Kodys, "Experimental Frailty Model towards an Adaptable Service Delivery for Aging People," in *21st International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 227–230, IEEE, nov 2016.

[13] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010.

[14] S. Tedesco, J. Barton, and B. O'Flynn, "A review of activity trackers for senior citizens: Research perspectives, commercial landscape and the role of the insurance industry," 2017.

[15] C. Felber, *Change everything : creating an economy for the common good*. 2015.

[16] "CO2 emissions calculator." https://www.sunearthtools.com/tools/CO2-emissions-calculator.php, 2017.

[17] Government of Singapore, "Personal Data Protection Act Overview."

[18] M. M. Theimer and B. N. Schilit, "Disseminating Active Map Information to Mobile Hosts," *IEEE Network*, vol. 8, no. 5, pp. 22–32, 1994.

[19] J. Bellmunt, M. Mokhtari, B. Abdulzarak, and H. Aloulou, "Agile framework for rapid deployment in ambient assisted living environments," in *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services - iiWAS '16*, (New York, New York, USA), pp. 410–413, ACM Press, 2016.

[20] "Sensors Overview — Android Developers."

[21] A. T. Barth, M. A. Hanson, H. C. Powell, J. Lach, and C. L. Brown, "Online data and execution profiling for dynamic energy-fidelity optimization in body sensor networks," in *2010 International Conference on Body Sensor Networks, BSN 2010*, 2010.

[22] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones," *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, no. April, pp. 24–26, 2013.

[23] M. Fowler, "Microservices."